Pentium® II Processor Specification Update

Release Date: December 1997

Order Number: 243337-008

The Pentium® II processor may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are documented in this Specification Update.

CONTENTS

REVISION HISTORY..... v
PREFACE..... vi
Specification Update for Pentium® II Processors
GENERAL INFORMATION 3
ERRATA 9
DOCUMENTATION CHANGES 32
SPECIFICATION CLARIFICATIONS 33
SPECIFICATION CHANGES..... 35

REVISION HISTORY

Date of Revision	Version	Description
May 1997	-001	This document is the first Specification Update for the Pentium® II processor.
June 1997	-002	Added Erratum 25. Update Erratum 13 status in the Summary Table of Changes. Added Documentation Change Table and Documentation Change 1. Added 300-MHz Pentium II processor information.
July 1997	-003	Added Erratum 26. Added Specification Change Table and Specification Changes 1 and 2.
August 1997	-004	Added Erratum 27. Added Document Change 2 and Spec Changes 3, 4, 5, 6, and 7.
September 1997	-005	Updated Erratum 27. Added Errata 28 and 29. Added Document Change 3 and Spec Clarification 1. Added C1 stepping information. Updated Specification Change 6.
October 1997	-006	Updated Errata 6 and 18, and S-spec table.
November 1997	-007	Updated Erratum 22. Added Specification Clarification 2, 3, and 4.
December 1997	-008	Updated and added notes to S-spec table. Updated package information table. Updated Erratum 24. Added Errata 30, 31, and 32.

PREFACE

This document is an update to the specifications contained the *Pentium® II Processor Developer's Manual* (Order Number 243341), the *Pentium® II Processor* datasheet and the *Intel Architecture Software Developer's Manual*, *Volumes 1, 2 and 3* (Order Numbers 243190, 243191, and 243192, respectively). It is intended for hardware system manufacturers and software developers of applications, operating systems, or tools. It contains Specification Changes, S-Specs, Errata, Specification Clarifications, and Documentation Changes.

Nomenclature

Specification Changes are modifications to the current published specifications for the Pentium® II processor. These changes will be incorporated in the next release of the specifications.

S-Specs are exceptions to the published specifications, and apply only to the units assembled under that s-spec.

Specification Clarifications describe a specification in greater detail or further highlight a specification's impact to a complex design situation. These clarifications will be incorporated in the next release of the specifications.

Documentation Changes include typos, errors, or omissions from the current published specifications. These changes will be incorporated in the next release of the specifications.

Errata are design defects or errors. Errata may cause the Pentium II processor's behavior to deviate from published specifications. Hardware and software designed to be used with any given processor stepping must assume that all errata documented for that processor stepping are present on all devices.

Identification Information

The Pentium II processor can be identified by the following values:

Family[1]	233-, 266-, 300-MHz Model 3[2]
0110	0011

NOTES:

1. The Family corresponds to bits [11:8] of the EDX register after RESET, bits [11:8] of the EAX register after the CPUID instruction is executed with a 1 in the EAX register, and the generation field of the Device ID register accessible through Boundary Scan.
2. The Model corresponds to bits [7:4] of the EDX register after RESET, bits [7:4] of the EAX register after the CPUID instruction is executed with a 1 in the EAX register, and the model field of the Device ID register accessible through Boundary Scan.

The Pentium II processor's second level (L2) cache size can be determined by the following register contents:

512-Kbyte Unified L2 Cache[1]	43h

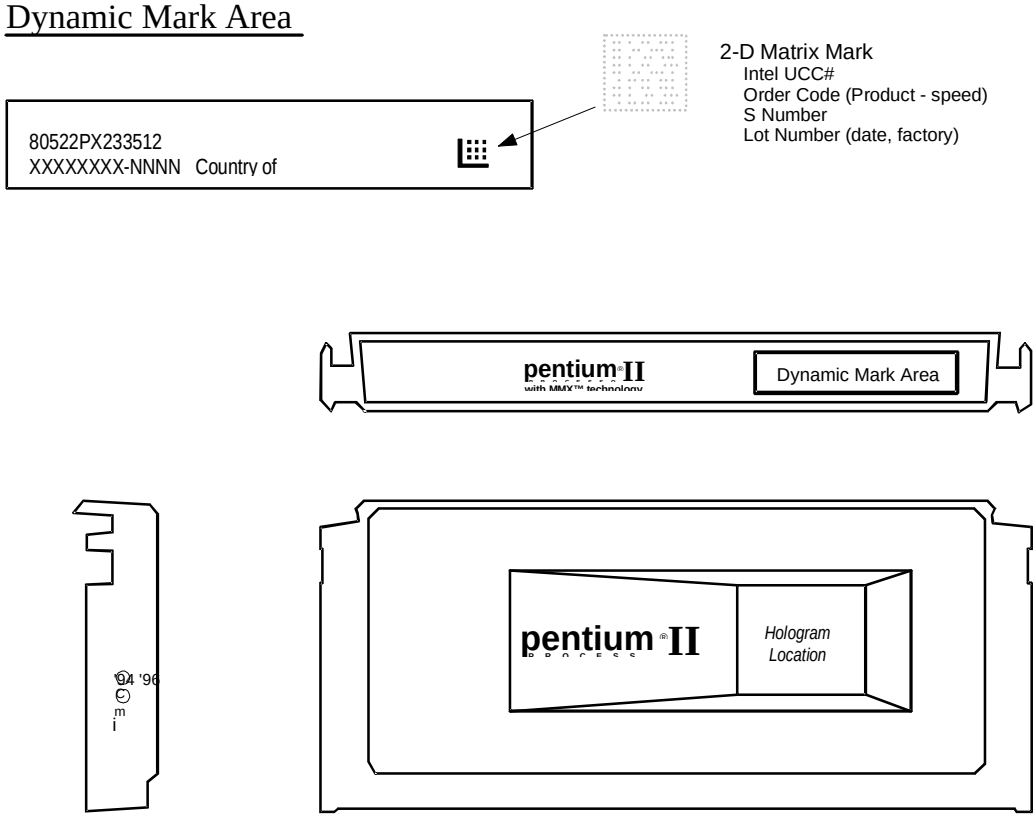
NOTE:

1. For the Pentium® II processor, the unified L2 cache size corresponds to the value in bits [3:0] of the EDX register after the CPUID instruction is executed with a 2 in the EAX register. Other Intel microprocessor models or families may move this information to other bit positions or otherwise reformat the result returned by this instruction; generic code should parse the resulting token stream according to the definition of the CPUID instruction.

Specification Update for Pentium® II Processors

GENERAL INFORMATION

Pentium® II Processor Markings



Basic Pentium® II Processor Identification Information

CPUID				Core Stepping	L1 Size (Kbytes)	T6 (81239AB) TagRAM Stepping	S-Spec	ECC/ Non-ECC	Speed (MHz) Core/Bus	Notes
Type	Family	Model	Stepping							
0	6	3	3	C0	512	B0	SL264	non-ECC	233/66	1, 2
0	6	3	3	C0	512	B0	SL265	non-ECC	266/66	1, 2
0	6	3	3	C0	512	B0	SL268	ECC	233/66	1, 2
0	6	3	3	C0	512	B0	SL269	ECC	266/66	1, 2
0	6	3	3	C0	512	B0	SL28K	non-ECC	233/66	1, 2, 3
0	6	3	3	C0	512	B0	SL28L	non-ECC	266/66	1, 2, 3
0	6	3	3	C0	512	B0	SL28R	ECC	300/66	1, 2
0	6	3	3	C0	512	B0	SL2MZ	ECC	300/66	1, 2, 3
0	6	3	3	C0	512	B0	SL2PV	ECC	266/66	1, 2, 3
0	6	3	4	C1	512	B0	SL2HA	ECC	300/66	1, 2
0	6	3	4	C1	512	B0	SL2HC	non-ECC	266/66	1, 2
0	6	3	4	C1	512	B0	SL2HD	non-ECC	233/66	1, 2
0	6	3	4	C1	512	B0	SL2HE	ECC	266/66	1, 2
0	6	3	4	C1	512	B0	SL2HF	ECC	233/66	1, 2
0	6	3	4	C1	512	B0	SL2QA	non-ECC	233/66	1, 2, 3
0	6	3	4	C1	512	B0	SL2QB	non-ECC	266/66	1, 2, 3
0	6	3	4	C1	512	B0	SL2QC	ECC	300/66	1, 2, 3
0	6	3	4	C1	512	B0	SL2QD	ECC	266/66	1, 2, 3

NOTES:

1. V_{CC_CORE} is specified for 2.8V +100/-70 mV for all Pentium® II processors.
2. T_{PLATE} is specified for 5°C - 75°C for all Pentium II processors with S.E.C. cartridge packages except for 300MHz Pentium II processors (SL28R , SL2HA, SL2MZ, and SL2QC which have a Tplate specification for 5ºC - 72ºC).
3. This is a boxed Pentium II processor with an attached fan heatsink.

Pentium® II Processor Package Information

S-Spec Number	Core			Processor Substrate Revision	Cartridge Revision
	Stepping	Speed (MHz)	L2 Size(Kbytes)		
SL264	C0	233	512	D	3.00
SL265	C0	266	512	D	3.00
SL268	C0	233	512	D	3.00
SL269	C0	266	512	D	3.00
SL28R	C0	300	512	D	3.00
SL2HD	C1	233	512	D	3.00
SL2HC	C1	266	512	D	3.00
SL2HF	C1	233	512	D	3.00
SL2HE	C1	266	512	D	3.00
SL2HA	C1	300	512	D	3.00
SL28K	C0	233	512	D	3.00
SL28L	C0	266	512	D	3.00
SL2MZ	C0	300	512	D	3.00
SL2QA	C1	233	512	D	3.00
SL2QB	C1	266	512	D	3.00
SL2QC	C1	300	512	D	3.00
SL2PV	C0	266	512	D	3.00
SL2QD	C1	266	512	D	3.00

Summary Table of Changes

The following table indicates the Specification Changes, Errata, Specification Clarifications, or Documentation Changes which apply to the Pentium II processors. Intel intends to fix some of the errata in a future stepping of the component, and to account for the other outstanding issues through documentation or specification changes as noted. This table uses the following notations:

CODES USED IN SUMMARY TABLE

X:	Specification Change, Erratum, Specification Clarification, or Documentation Change applies to the given processor stepping.
Doc:	Intel intends to update the appropriate documentation in a future revision.
Fix:	This erratum is intended to be fixed in a future stepping of the component.
Fixed:	This erratum has been previously fixed.
NoFix:	There are no plans to fix this erratum.
(No mark) or (blank box):	This item is fixed in or does not apply to the given stepping.
AP:	APIC related erratum.
SUB:	This column refers to errata on the Pentium® II processor substrate.
Shaded:	This erratum is either new or modified from the previous version of the document.

NO.	C0	C1	SUB	Plans	ERRATA
1	X	X		NoFix	FP Operand Pointer may be incorrectly calculated after FP access which wraps 64-Kbyte boundary in 16-bit code
2	X	X		NoFix	Differences exist in debug exception reporting
3	X	X		NoFix	FLUSH# servicing delayed while waiting for STARTUP_IPI in 2-way MP systems
4	X	X		NoFix	Code fetch matching disabled debug register may cause debug exception
5	X	X		NoFix	Double ECC error on read may result in BINIT#
6	X	X		NoFix	FP inexact-result exception flag may not be set
7	X	X		NoFix	BTM for SMI will contain incorrect FROM EIP
8	X	X		NoFix	I/O restart in SMM may fail after simultaneous MCE

NO.	C0	C1	SUB	Plans	ERRATA
9	X	X		NoFix	Branch traps do not function if BTMs are also enabled
10	X	X		NoFix	Checker BIST failure in FRC mode not signaled
11	X	X		NoFix	BINIT# assertion causes FRCERR assertion in FRC mode
12	X	X		NoFix	Machine Check Exception handler may not always execute successfully
13				Fixed	MCE due to L2 parity error gives L1 MCACOD.LL
14	X	X		NoFix	LBER may be corrupted after some events
15	X	X		NoFix	BTM's may be corrupted during simultaneous L1 cache line replacement
16	X			Fix	System may hang due to internal protocol violation
17	X			Fix	Livelock condition may cause system hang
18	X	X		Fix	Mispredicted branch may cause incorrect tag word on MMX™ technology instructions
19	X	X		Fix	Thermal sensor/THERMTRIP# does not work
20	X	X		Fix	Spurious machine check exception via IFU data parity error
21	X	X		Fix	Loss of inclusion in IFU can cause Machine Check Exception
22	X	X		Fix	Possible system hang when paging is disabled and re-enabled from uncached memory
23	X	X		Fix	L2 performance counters miscount L2_RQSTS
24	X	X		Fix	Erroneous signaling of user mode protection violation
25	X			Fix	Invalid operation not signaled by the FIST instruction on some out of range operands
26	X	X		Fix	FLUSH# assertion disables L2 Machine Check Exception reporting
27	X	X		Fix	EFLAGS may be incorrect after a multiprocessor TLB shootdown
28	X	X		No Fix	Delayed line invalidation issue during 2-way MP data ownership transfer
29	X	X		Fix	Potential early deassertion of LOCK# during split-lock cycles
30	X	X		NoFix	A20M# may be inverted after returning from SMM and Reset

NO.	C0	C1	SUB	Plans	ERRATA
31	X	X		Fix	Reporting of floating-point exception may be delayed
32	X	X		NoFix	EFLAGS discrepancy on a page fault after a multiprocessor TLB shootdown
1AP	X	X		NoFix	APIC access to cacheable memory causes SHUTDOWN
2AP	X	X		NoFix	2-way MP systems may hang due to catastrophic errors during BSP determination
3AP	X	X		NoFix	Write to mask LVT (programmed as EXTINT) will not deassert outstanding interrupt
NO.	**C0**	**C1**	**SUB**	**Plans**	**DOCUMENTATION CHANGES**
1	X	X		Doc	Note 4 of Table 13, note 6 of Table 16, Output Valid Delay specified to 2.5V +5% in datasheet
2	X	X		Doc	Invalid arithmetic operations and masked responses to them relative to FIST/FISTP instruction
3	X	X		Doc	FIDIV/FIDIVR m16int description
NO.	**C0**	**C1**	**SUB**	**Plans**	**SPECIFICATION CLARIFICATIONS**
1	X	X		Doc	Writes to WC memory
2	X	X		Doc	Multiple processors protocol and restrictions
3	X	X		Doc	NMI handling while in SMM
4	X	X		Doc	Critical sequence of events during a page fault exception
NO.	**C0**	**C1**	**SUB**	**Plans**	**SPECIFICATION CHANGES**
1	X	X		Doc	BCLK period stability change from ±250ps to ±300ps
2	X	X		Doc	$V_{CC_{CORE}}$ transient tolerance change by 5mV and 10mV
3	X	X		Doc	Deep Sleep ICC CORE Specification change from 0.2A to 0.35A
4	X	X		Doc	TRST# Pull-down Resistor
5	X	X		Doc	300 MHz Pentium® II Processor Maximum Temperature Change
6	X	X		Doc	S.E.C. Cartridge Mechanical Specifications
7	X	X		Doc	S.E.C. Cartridge Heatsink Attach Clearance Specification Change

ERRATA

1. FP Operand Pointer May Be Incorrectly Calculated After FP Access Which Wraps 64-Kbyte Boundary in 16-Bit Code

PROBLEM: The FP Operand Pointer is the effective address of the operand associated with the last non-control floating-point instruction executed by the machine. If an 80-bit floating-point access (load or store) occurs in a 16-bit mode other than protected mode (in which case the access will produce a segment limit violation), the memory access wraps a 64-Kbyte boundary, and the floating-point environment is subsequently saved in 32-bit mode, the subtraction routine used to calculate the FP Operand Pointer will assume the floating-point access was in 32-bit mode, and the high word of the address will be FFFFh instead of 0000h.
IMPLICATION: A 32-bit operating system running 16-bit floating-point code may encounter this erratum, under the following conditions:

- The operating system is using a segment greater than 64 Kbytes in size.
- An application is running in a 16-bit mode other than protected mode.
- An 80-bit floating-point load which wraps the 64-Kbyte boundary is executed.
- The operating system uses a 32-bit handler on an unmasked exception which occurs during the load.
- The exception handler uses the value contained in the FP Operand Pointer.

Wrapping an 80-bit floating-point load around a segment boundary in this way is not a normal programming practice. Intel has not currently identified any software which exhibits this behavior.
WORKAROUND: If the FP Operand Pointer is used in a 32-bit exception handler in an OS which may run 16-bit floating-point code, care must be taken to ensure that no 80-bit floating-point accesses are wrapped around a 64-Kbyte boundary.
STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

2. Differences Exist in Debug Exception Reporting

PROBLEM: There exist some differences in the reporting of code and data breakpoint matches between that specified by previous Intel processors' specifications and the behavior of the Pentium II processor, as described below:

CASE 1:
The first case is for a breakpoint set on a MOVSS or POPSS instruction, when the instruction following it causes a debug register protection fault (DR7.gd is already set, enabling the fault). The Pentium processor reports delayed data breakpoint matches from the MOVSS or POPSS

instructions by setting the matching DR6.bi bits, along with the debug register protection fault (DR6.bd). If additional breakpoint faults are matched during the call of the debug fault handler, the Pentium processor sets the breakpoint match bits (DR6.bi) to reflect the breakpoints matched by both the MOVSS or POPSS breakpoint and the debug fault handler call. The Pentium II processor only sets DR6.bd in either situation, and does not set any of the DR6.bi bits.

CASE 2:
In the second breakpoint reporting failure case, if a MOVSS or POPSS instruction with a data breakpoint is followed by a store to memory which crosses a 4-Kbyte page boundary, the breakpoint information for the MOVSS or POPSS will be lost. Previous processors retain this information across such a page split.

CASE 3:
If they occur after a MOVSS or POPSS instruction, the INT *n*, INTO, and INT3 instructions zero the DR6.Bi bits (bits B0 through B3), clearing pending breakpoint information, unlike previous processors.

CASE 4:
If a data breakpoint and an SMI (System Management Interrupt) occur simultaneously, the SMI will be serviced via a call to the SMM handler, and the pending breakpoint will be lost.
IMPLICATION: When debugging or when developing debuggers for a Pentium II processor based system, this behavior should be noted. Normal usage of the MOVSS or POPSS instructions (i.e., following them with a MOV ESP) will not exhibit the behavior of cases 1-3. Debugging in conjunction with SMM will be limited by case 4 (no workaround has been identified for this case).
WORKAROUND: Following MOVSS and POPSS instructions with a MOV ESP instruction when using breakpoints will avoid the first three cases of this erratum.
STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

3. FLUSH# Servicing Delayed While Waiting for STARTUP_IPI in 2-way MP Systems

PROBLEM: In a 2-way MP system, if an application processor is waiting for a startup interprocessor interrupt (STARTUP_IPI), then it will not service a FLUSH# pin assertion until it has received the STARTUP_IPI.
IMPLICATION: After the 2-way MP initialization protocol, only one processor becomes the bootstrap processor (BSP). The other processor becomes a slave application processor (AP). After losing the BSP arbitration, the AP goes into a wait loop, waiting for a STARTUP_IPI. The BSP can wake up the AP to perform some tasks with a STARTUP_IPI, and then put it back to sleep with an initialization interprocessor interrupt (INIT_IPI, which has the same affect as

asserting INIT#), which returns it to a wait loop. The result is a possible loss of cache coherency if the offline processor is intended to service a FLUSH# assertion at this point. The FLUSH# will be serviced as soon as the processor is awakened by a STARTUP_IPI, before any other instructions are executed. Intel has not encountered any operating systems that are affected by this erratum.

WORKAROUND: Operating system developers should take care to execute a WBINVD instruction before the AP is taken offline using an INIT_IPI.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

4. Code Fetch Matching Disabled Debug Register May Cause Debug Exception

PROBLEM: The bits L0-3 and G0-3 enable breakpoints local to a task and global to all tasks, respectively. If one of these bits is set, a breakpoint is enabled, corresponding to the addresses in the debug registers DR0-DR3. If at least one of these breakpoints is enabled, any of these registers are *disabled* (i.e., L*n* and G*n* are 0), and RW*n* for the disabled register is 00 (indicating a breakpoint on instruction execution), normally an instruction fetch will not cause an instruction-breakpoint fault based on a match with the address in the disabled register (s). However, if the address in a disabled register matches the address of a code fetch which also results in a page fault, an instruction-breakpoint fault will occur.

IMPLICATION: While debugging software, extraneous instruction-breakpoint faults may be encountered if breakpoint registers are not cleared when they are disabled. Debug software which does not implement a code breakpoint handler will fail, if this occurs. If a handler is present, the fault will be serviced. Mixing data and code may exacerbate this problem by allowing disabled data breakpoint registers to break on an instruction fetch.

WORKAROUND: The debug handler should clear breakpoint registers before they become disabled.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

5. Double ECC Error on Read May Result in BINIT#

PROBLEM: For this erratum to occur, the following conditions must be met:

- Machine Check Exceptions (MCEs) must be enabled.
- A dataless transaction (such as a write invalidate) must be occurring simultaneously with a transaction which returns data (a normal read).
- The read data must contain a double-bit uncorrectable ECC error.

If these conditions are met, the Pentium II processor will not be able to determine which transaction was erroneous, and instead of generating an MCE, it will generate a BINIT#.

IMPLICATION: The bus will be reinitialized in this case. However, since a double-bit uncorrectable ECC error occurred on the read, the MCE handler (which is normally reached on a double-bit uncorrectable ECC error for a read) would most likely cause the same BINIT# event.
WORKAROUND: Though the ability to drive BINIT# can be disabled in the Pentium II processor, which would prevent the effects of this erratum, overall system behavior would not improve, since the error which would normally cause a BINIT# would instead cause the machine to shut down. No other workaround has been identified.
STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

6. FP Inexact-Result Exception Flag May Not Be Set

PROBLEM: When the result of a floating-point operation is not exactly representable in the destination format (1/3 in binary form, for example), an inexact-result (precision) exception occurs. When this occurs, the PE bit (bit 5 of the FPU status word) is normally set by the processor. Under certain rare conditions, this bit may not be set when this rounding occurs. However, other actions taken by the processor (invoking the software exception handler if the exception is unmasked) are not affected. This erratum can only occur if the floating-point operation which causes the precision exception is immediately followed by one of the following instructions:

- FST m32real
- FST m64real
- FSTP m32real
- FSTP m64real
- FSTP m80real
- FIST m16int
- FIST m32int
- FISTP m16int
- FISTP m32int
- FISTP m64int

Note that even if this combination of instructions is encountered, there is also a dependency on the internal pipelining and execution state of both instructions in the processor.
IMPLICATION: Inexact-result exceptions are commonly masked or ignored by applications, as it happens frequently, and produces a rounded result acceptable to most applications. The PE bit of the FPU status word may not always be set upon receiving an inexact-result exception. Thus, if these exceptions are unmasked, a floating-point error exception handler may not recognize that a precision exception occurred. Note that this is a “sticky” bit, i.e., once set by an inexact-result condition, it remains set until cleared by software.

WORKAROUND: This condition can be avoided by inserting a NOP instruction between the two floating-point instructions.
STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

7. BTM for SMI Will Contain Incorrect FROM EIP

PROBLEM: A system management interrupt (SMI) will produce a Branch Trace Message (BTM), if BTMs are enabled. However, the FROM EIP field of the BTM (used to determine the address of the instruction which was being executed when the SMI was serviced) will not have been updated for the SMI, so the field will report the same FROM EIP as the previous BTM.
IMPLICATION: A BTM which is issued for an SMI will not contain the correct FROM EIP, limiting the usefulness of BTMs for debugging software in conjunction with System Management Mode (SMM).
WORKAROUND: None identified at this time.
STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

8. I/O Restart in SMM May Fail After Simultaneous MCE

PROBLEM: If an I/O instruction (IN, INS, REP INS, OUT, OUTS, or REP OUTS) is being executed, and if the data for this instruction becomes corrupted, the Pentium II processor will signal a machine check exception (MCE). If the instruction is directed at a device which is powered down, the processor may also receive an assertion of SMI#. Since MCEs have higher priority, the processor will call the MCE handler, and the SMI# assertion will remain pending. However, upon attempting to execute the first instruction of the MCE handler, the SMI# will be recognized and the processor will attempt to execute the SMM handler. If the SMM handler is completed successfully, it will attempt to restart the I/O instruction, but will not have the correct machine state, due to the call to the MCE handler.
IMPLICATION: A simultaneous MCE and SMI# assertion may occur for one of the I/O instructions above. The SMM handler may attempt to restart such an I/O instruction, but will have corrupted state due to the MCE handler call, leading to failure of the restart and SHUTDOWN of the processor.
WORKAROUND: If a system implementation must support both SMM and MCEs, the first thing the SMM handler code (when an I/O restart is to be performed) should do is check for a pending MCE. If there is an MCE pending, the SMM handler should immediately exit via an RSM instruction and allow the MCE handler to execute and restart the I/O instruction. If there is not, the SMM handler may proceed with its normal operation.
STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

9. Branch Traps Do Not Function if BTMs Are Also Enabled

PROBLEM: If branch traps or branch trace messages (BTMs) are enabled alone, both function as expected. However, if both are enabled, only the BTMs will function, and the branch traps will be ignored.
IMPLICATION: The branch traps and branch trace message debugging features cannot be used together.
WORKAROUND: If branch trap functionality is desired, BTMs must be disabled.
STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

10. Checker BIST Failure in FRC Mode Not Signaled

PROBLEM: If a system is running in functional redundancy checking (FRC) mode, and the checker of the master-checker pair encounters a hard failure while running the built-in self test (BIST), the checker will tri-state all outputs without signaling an IERR#.
IMPLICATION: Assuming the master passes BIST successfully, it will continue execution unchecked, operating without functional redundancy. However, the necessary pull-up on the FRCERR pin will cause an FRCERR to be signaled. The operation of the master depends on the implementation of FRCERR.
WORKAROUND: For successful detection of BIST failure in the checker of an FRC pair, use the FRCERR signal, instead of IERR#.
STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

11. BINIT# Assertion Causes FRCERR Assertion in FRC Mode

PROBLEM: If a pair of Pentium II processors are running in functional redundancy checking (FRC) mode, and a catastrophic error condition causes BINIT# to be asserted, the checker in the master-checker pair will enter SHUTDOWN. The next bus transaction from the master will then result in the assertion of FRCERR.
IMPLICATION: Bus initialization via an assertion of BINIT# occurs as the result of a catastrophic error condition which precludes the continuing reliable execution of the system. Under normal circumstances, the master-checker pair would remain synchronized in the execution of the BINIT# handler. However, due to this erratum, an FRCERR will be signaled. System behavior then depends on the behavior of the system specific error recovery mechanisms.
WORKAROUND: None identified at this time.
STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

12. Machine Check Exception Handler May Not Always Execute Successfully

PROBLEM: An asynchronous machine check exception (MCE), such as a BINIT# event, which occurs during an access that splits a 4-Kbyte page boundary may leave some internal registers in an indeterminate state. Thus, MCE handler code may not always run successfully if an asynchronous MCE has occurred previously.
IMPLICATION: An MCE may not always result in the successful execution of the MCE handler. However, asynchronous MCEs usually occur upon detection of a catastrophic system condition that would also hang the processor. Leaving MCEs disabled will result in the condition which caused the asynchronous MCE instead causing the processor to enter SHUTDOWN. Therefore, leaving MCEs disabled may not improve overall system behavior.
WORKAROUND: No workaround which would guarantee successful MCE handler execution under this condition has been identified.
STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

13. MCE Due to L2 Parity Error Gives L1 MCACOD.LL

PROBLEM: If a Cache Reply Parity (CRP) error, Cache Address Parity (CAP) error, or Cache Synchronous Error (CSER) occurs on an access to the Pentium II processor's L2 cache, the resulting Machine Check Architectural Error Code (MCACOD) will be logged with '01' in the LL field. This value indicates an L1 cache error; the value should be '10', indicating an L2 cache error. Note L2 ECC errors have the correct value of '10' logged.
IMPLICATION: An L2 cache access error, other than an ECC error, will be improperly logged as an L1 cache error in MCACOD.LL.
WORKAROUND: None identified at this time.
STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

14. LBER May Be Corrupted After Some Events

PROBLEM: The last branch record (LBR) and the last branch before exception record (LBER) can be used to determine the source and destination information for previous branches or exceptions. The LBR contains the source and destination addresses for the last branch or exception, and the LBER contains similar information for the last branch taken before the last exception. This information is typically used to determine the location of a branch which leads to execution of code which causes an exception. However, after a catastrophic bus condition which results in an assertion of BINIT# and the reinitialization of the buses, the value in the LBER may be corrupted. Also, after either a CALL which results in a fault or a software interrupt, the LBER and LBR will be updated to the same value, when the LBER should not have been updated.

IMPLICATION: The LBER and LBR registers are used only for debugging purposes. When this erratum occurs, the LBER will not contain reliable address information. The value of LBER should be used with caution when debugging branching code; if the values in the LBR and LBER are the same, then the LBER value is incorrect. Also, the value in the LBER should not be relied upon after a BINIT# event.
WORKAROUND: None identified at this time.
STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

15. BTMs May Be Corrupted During Simultaneous L1 Cache Line Replacement

PROBLEM: When Branch Trace Messages (BTMs) are enabled and such a message is generated, the BTM may be corrupted when issued to the bus by the L1 cache if a new line of data is brought into the L1 data cache simultaneously. Though the new line being stored in the L1 cache is stored correctly, and no corruption occurs in the data, the information in the BTM may be incorrect due to the internal collision of the data line and the BTM.
IMPLICATION: Although BTMs may not be entirely reliable due to this erratum, the conditions necessary for this boundary condition to occur have only been exhibited during focused simulation testing. Intel has currently not observed this erratum in a system level validation environment.
WORKAROUND: None identified at this time.
STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

16. System May Hang Due To Internal Protocol Violation

PROBLEM: Pentium II processor based systems may hang due to an internal protocol violation. When a snoopable transaction is issued on the bus and the cache line being accessed is in the modified state, the processor must deliver to the system bus an updated copy of the cache line. When the processor attempts to deliver the most up to date copy via an implicit writeback, the data transfer transaction fails and the DBSY# signal remains asserted until the next RESET#. This causes the system to hang indefinitely. In order to encounter this erratum, the following sequence of events must occur:

1. A snoopable transaction (transaction 1) is issued on the system bus. The processor contains in its L1 and/or L2 caches the data for this line in the modified state.
2. Another snoopable transaction (transaction 2) is issued and the processor contains this line only in its L2 cache in the modified state. Both of these transactions can be issued by either the chipset, by the processor (in which case they are of the self-snoop type), by another processor (2-way MP systems), or any combination thereof.
3. A non-snoopable transaction is then issued (transaction 3) for which address bits A15-A5 are the same as those in transaction 2.
4. Transaction 3 is followed by a snoopable transaction (transaction 4).

5. The completion of the data transfer phase of transaction 1 must line up with the snoop response phase of transaction 3. This data transfer phase of transaction 1 must occur after the ADS# of transaction 4 and line up with the completion of an internal cache transaction.
6. The internal cache transaction must miss the L2 targeting a line for eviction, but the internal cache transaction must be such that it has to be re-tried.

The result of this sequence of transactions causes the processor bus to lock up after delivering the data for transaction 1, but prior to delivering the data for transaction 2. Since this data is never delivered, DBSY# does not deassert and the system hangs.
IMPLICATION: The Pentium II processor may cause a system to hang if the above listed sequence of events occurs. This sequence is a necessary condition to hit the erratum, but multiple variations of this sequence which also cause this erratum are also possible. The probability of encountering this erratum increases with I/O queue depth greater than 4 and in 2-way MP systems.
WORKAROUND: None identified at this time.
STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

17. Livelock Condition May Cause System Hang

PROBLEM: A “livelock” situation could occur in 2-way MP Pentium II processor-based systems, when IOQ depth is set to 1, with a failure signature such that a processor arbitrates for the system bus but fails to drive out a transaction when it gains ownership of the bus. The processor then relinquishes bus ownership to another requester, but on re-arbitration performs the same repetitive actions. This course of action continues until RESET# is asserted. The failure signature in 2-way MP systems is such that both processors require execution of an explicit writeback cycle and both processors request the bus for this transaction. However, when the time comes to drive out the writeback transaction, the internal request has been suspended due to an internal blocking condition. After the internal blocking condition has gone away the original writeback request is re-asserted. However, by the time bus ownership has been re-gained, the blocking condition has recurred, thus suppressing the writeback request before the transaction can be driven out to the system bus.
The writeback which is waiting to go out on the system bus must be issued before the internal blocking condition can be removed. But the writeback can never be issued because of the recurring blocking condition. This causes an “infinite loop” situation to develop, and the processor essentially stops executing code.
IMPLICATION: This erratum was observed to occur when both processors are configured for IOQ depth = 1 in Intel commercial system testing.
WORKAROUND: None identified at this time.
STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

18. Mispredicted Branch May Cause Incorrect Tag Word on MMX™ Technology Instructions

PROBLEM: After any MMX™ technology instruction is executed, all of the FPU stack registers should be marked valid in the FPU tag word. If one or more of the first three instructions of a mispredicted branch are MMX technology instructions of the form “opcode reg, mem” not including MOVD and MOVQ, the FPU tag word is incorrectly modified. Some of the tag word bits may remain invalid. This tag word will remain incorrect until one of two events occur:

1. Any MMX™ technology instruction is executed four or more instructions after the branch target, or
2. An MMX technology instruction of the following type is executed:
   - Any MMX technology instruction of the form “opcode reg, reg”
   - MOVD
   - MOVQ
   - EMMS

The following are examples of code that will encounter this erratum.

Example 1:

```
        EMMS
        ...
        Jcc     target          ; mispredicted as not taken
        ...
target:
        PADDW   mm0, [edi]      ; Is an “reg, mem” format instruction
        FSTENV  env
```

In this example, the tag word stored in memory by FSTENV will be incorrect.

Example 2:

```
        EMMS
        ...
        Jcc     target          ; mispredicted as not taken
        ...
target:
        PADDW   mm0, [edi]
        FUCOMPP                 ; depends on tag word, also violates coding guideline against
        mixing
                                ; floating-point and MMX™ technology instructions
        FWAIT
```

In this example, the FUCOMPP instruction will cause a Numeric Invalid Operation Exception if the FPU stack fault exception is unmasked.

IMPLICATION: When writing code that mixes FP and MMX technology instructions where the target of a branch is an MMX technology instruction with a memory operand, the FPU tag word may be incorrect. Software that expects the FP stack register to be set to valid after an MMX technology instruction and utilizes this information may be affected.

If floating-point instructions are intermixed, the floating-point instructions may raise the floating-point stack exception. If this exception is unmasked, the application will receive an unexpected numeric exception. The result is application dependent. If the floating-point stack exception is masked, the floating-point instruction will compute with a indefinite operand instead of the register contents. In either case the result is application dependent. Applications that follow the Intel MMX Technology Coding Guidelines against intermixing floating-point and MMX technology code are not affected by this erratum.

If the floating-point tag word is saved immediately after an affected MMX technology instruction, an erroneous value will be stored. Program behavior is application dependent. This may also cause debuggers to temporarily display incorrect tag word contents.

WORKAROUND: All of the following must be applied to work around this erratum:

- Follow the Intel MMX™ technology guidelines in the *Intel Architecture Developer's Optimization Manual* for writing MMX technology programs. Specifically, do not intermix MMX technology instructions and floating-point instructions on a per instruction basis.
- If it is possible that some of the tag word bits may be invalid prior to a branch, avoid using MMX technology instructions of the form "opcode reg, mem", except MOVD, MOVQ, within the first three instructions at the target of a branch.
- Use the FSAVE instruction to save all floating-point stack registers if at least one of the registers is valid during a context switch.
- Before a nonstandard transition from MMX technology code to floating-point code, execute a non-susceptible MMX technology instruction such as MOVD eax, mm0.
- Floating point instructions should not depend on MMX technology instructions to set the tag word bits to valid.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

19. Thermal Sensor/THERMTRIP# Does Not Work

PROBLEM: THERMTRIP# is a feature of the Pentium II processor which asserts when the core reaches a certain temperature during operation as specified in the *Pentium® II Processor* datasheet. The Pentium II processor may assert THERMTRIP# at a temperature lower or higher than the specified trippoint of 135°C for $T_{JUNCTION}$. When THERMTRIP# is asserted, the processor may shut down causing all execution to be halted.

IMPLICATION: When running the Pentium II processor, the Pentium II processor core may reach a temperature causing the processor to assert THERMTRIP# early. Once THERMTRIP# has been asserted, the processor may shut down due to this erratum. All execution after the SHUTDOWN will be halted. This erratum is only exhibited when T_{PLATE} is above the Maximum Specification of 75°C (see the *Pentium® II Processor* datasheet, Order Number 243335, for details on specifications).
WORKAROUND: Avoid operation of the Pentium II processor outside of thermal specifications defined by the *Pentium® II Processor* datasheet. Do not monitor the THERMTRIP# pin (pin A15).
STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

20. Spurious Machine Check Exception Via IFU Data Parity Error

PROBLEM: The Pentium II processor can signal an unrecoverable Machine Check Exception (MCE) in the event that the Instruction Fetch Unit (IFU) detects a mismatch when verifying instruction parity. The execution of code which modifies the current instruction sequence that may already be fetched into the processor can cause an instruction at a given address to appear differently depending on when it was fetched in time relative to its being modified. Thus, a speculatively prefetched instruction may have been modified such that it now differs from the copy of the same instruction resident in the instruction cache. This discrepancy (of one copy located in the speculative prefetch portion, and a different copy in the instruction cache) is sensed by the IFU. When the IFU detects that the instruction stream has been modified, it flushes the pipeline and attempts to restart the instruction stream. In the interim, the IFU recognizes the disparate instructions described above, and signals a data parity error. The data parity error is signaled as an MCE before the instruction stream has had a chance to restart. This MCE will cause an operating system that has enabled MCE to shut down. No incorrect code is executed by the processor in this situation (even if MCE is disabled). Note that this erratum occurs under a specific set of address dependencies and timing events.
IMPLICATION: Executing such a sequence by modifying code without proper synchronization may not always result in predictable program behavior. The processor's signaling of an MCE due to a data parity error in the IFU may then result in an unexpected system halt if the above conditions are met and MCEs are enabled.
WORKAROUND: It is possible for BIOS code to contain a workaround for this erratum.
STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

21. Loss of Inclusion In IFU Can Cause Machine Check Exception

PROBLEM: The Pentium II processor can signal an unrecoverable Machine Check Exception (MCE) as a consistency checking mechanism in the event that the Instruction Fetch Unit (IFU) detects differences in the consistency of code in instruction streaming buffers against code

resident in the instruction cache, i.e., a loss of inclusion. When application code makes an operating system call, the processor transitions execution privilege levels. If the code for the OS call is not already resident in the level 1 cache, then the processor may prefetch code while identifying a cache line(s) for eventual eviction to make space for the new code. Upon return from the OS call, the processor continues execution of application code at the user level. The processor, due to deep speculation and branch prediction, may attempt to execute instructions from the previously prefetched kernel code starting by attempting to replace the victim line with kernel code in a buffer internal to the IFU. The IFU detects that the current application is insufficiently privileged to execute the kernel code and so, suppresses the eviction of the previously selected victim line. Despite having detected this condition, the IFU does replace this victim line with the kernel line. If the processor now attempts to restart execution of the current application code by refetching the original victim line it no longer finds it in the instruction cache. The IFU detects this loss of inclusion, and signals this by generating a MCE. If MCEs are enabled, this event can cause an operating system to shut down. Note that this erratum occurs under a specific set of address dependencies and timing events.

IMPLICATION: The occurrence of all the conditions above can lead the IFU to signal a loss of inclusion by generating an MCE. If MCEs are enabled in the system, then the operating system may shut down upon noticing the MCE resulting in system failure. If MCEs are disabled, then unpredictable application behavior is theoretically possible, although current validation has shown execution to continue normally.

WORKAROUND: It is possible for BIOS code to contain a workaround for this erratum.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

22. Possible System Hang When Paging Is Disabled and Re-enabled from Uncached Memory

PROBLEM: If paging is disabled via the PG bit of CR0 and then later re-enabled while executing code from a page marked uncachable by its Page Table Entry (PCD=1) but located in memory mapped as Write Back or Write Through by the processor MTRRs, the processor could internally enter a state resulting in a system hang.

IMPLICATION: Operating systems that enable and disable paging with the above described memory configurations could hang. Intel has not observed this erratum to date in laboratory testing of commercially available operating systems and applications.

WORKAROUND: It is possible for BIOS code to contain a workaround for this erratum.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

23. L2 Performance Counters Miscount L2_RQSTS

PROBLEM: L2_RQSTS is a performance counter that counts the number of L2 cache access requests. This counter increments for each incoming L2 cache request. In some cases, an L2

request cannot be serviced by the L2 Cache. This request is then retried at a later time when the request can be serviced by the L2 cache. When this happens, the L2_RQSTS counter counts the initial L2 cache request **and** the retried L2 cache request, thereby counting the same request twice.

IMPLICATION: The L2_RQSTS counter may contain a larger erroneous number of L2 cache requests due to this erratum. This erratum does not affect functionality of the Pentium II processor. This erratum only affects the performance counter specified.

WORKAROUND: None identified at this time.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

24. Erroneous Signaling of User Mode Protection Violation

PROBLEM: If the Pentium II processor attempts to access a page in physical memory marked not present (Present bit clear), a page fault exception (#PF) is generated. Before proceeding, there is a narrow internal timing window where the processor verifies that no other higher priority fault conditions are present. During this time, it is possible for another agent to allocate a new page directory or page table entry (PDE/PTE) corresponding to the same linear address of the original access, writing new values into the PDE/PTE with the Access bit (A-bit) or the Dirty bit (D-bit) cleared. When the original processor completes its checking for other fault conditions, and re-examines the A/D bit of the recently modified PDE/PTE, it finds that it has been cleared. Internal hardware correctly signals this scenario as a condition to which the processor should respond by setting the A/D bit, but erroneously reports it as a generic paging protection violation. Instead of attempting to set the appropriate A/D bit, this event is reported as an Int14 with exception code 0x05, i.e., user mode protection violation.

IMPLICATION: The occurrence of this scenario will result in the erroneous signaling of a user mode protection violation instead of a page fault and may result in application termination depending on operating system behavior in response to a user mode protection violation. Intel has only observed this erratum to date in laboratory testing of multi-processor systems.

WORKAROUND: Operating systems which allocate new PTEs and PDEs should set the Access bit (A-bit) and Dirty bit (D-bit) to work around this erratum. Alternately, an operating system's Int14 handler can determine if a protection violation condition truly exists, and if none is found, return without further action.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

25. Invalid Operation Not Signaled By The FIST Instruction on Some Out of Range Operands

PROBLEM: On certain, large, negative, floating-point operands, and only in three of the four possible processor rounding modes, the instructions FIST[P] m16int and FIST[P] m32int do not detect that the operand is so large that it will not fit into the target data size. As a consequence,

the expected Invalid Operation exception response for this situation is not correctly provided, nor is the Invalid Operation flag set in the Floating Point status word as specified in the *Intel Architecture Programmers Reference Manual, Volume 2*. Under the failing conditions, noted below, the precision exception (#PE) will also be incorrectly set.
The erratum occurs only when all of the following conditions are met:

1. Operations are unaffected.
2. The FIST[P] instruction is either a 16- or 32-bit operation; 64-bit Either the '*to nearest*', '*to zero*' or '*up*' rounding modes are being used. The round '*down*' mode is unaffected by this erratum.
3. The sign bit of the floating-point operand is negative.
4. The floating-point operand being converted is significantly more negative than can be described by the integer size being targeted.

ACTUAL vs. EXPECTED RESPONSE
A. Actual Response:
When the required conditions are encountered, the processor provides the following response:

- Return the MAXNEG value (8000h for FIST16 & 80000000h for FIST32) to memory.
- The IE (Invalid Operation) bit in the Floating Point status word is *not set* to flag the use of an invalid operand.
- The PE (precision error) bit the Floating Point status word is *set*.
- No exception handler is invoked.
- In the case of a FISTP instruction the Operand will have been popped from the floating point stack

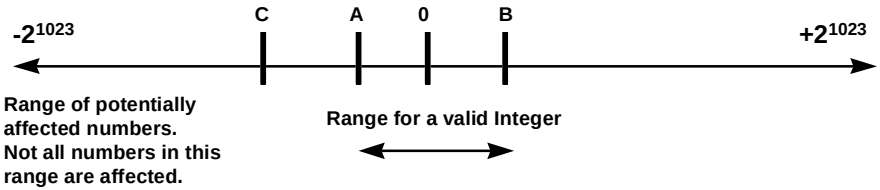
B. Expected Response
The expected processor response when the invalid operation exception is *masked* is:

- Return the MAXNEG value (8000h for FIST16 & 80000000h for FIST32) to memory.
- The IE (Invalid Operation) bit in the Floating Point status word is *set* to flag the overflow.
- The PE (precision error) bit the Floating Point status word is *not set*.

The expected processor response when the invalid operation exception is *unmasked* is:

- Do not return a result to memory. Keep the original operand intact on the stack.
- The IE (Invalid Operation) bit in the Floating Point status word is *set* to flag the overflow.
- The PE (precision error) bit the Floating Point status word is *not set*.
- Vector to the user numeric exception handler.

IMPLICATION: Erroneous operation results when the operand is so large that it will not fit into the target data size. The operands affected by this erratum are significantly outside (by a factor of 3X) the range that can be, correctly, converted to an integer value. The figure below and corresponding table identifies the normal range of integer numbers (between A and B) and the starting point of the operands affected by this erratum. Discrete failing operands will be present in the range between point C and the maximum negative number that can be represented by the processor (-2^{1023} in double precision format). Note 2 below gives a qualitative description of the nature of the discrete failing values. Software that does not rely on the Invalid Operation exception flag being set and signaled by either an exception OR by software polling is not impacted by this erratum.



16-bit Operation	A	B	C
	-32,768.0	+32,767.0	< -98304.0
32-bit Operation	A	B	C
	-2,147,483,648.0	+2,147,483,647.0	< -6,442,450,944.0

WORKAROUND: Any of two software workarounds will avoid occurrence of this erratum:

1. Range checking performed prior to execution of the FIST[P] instruction will prevent the overflow condition from occurring, and may already be implemented as a standard coding style.
2. Software can use the presence of MAXNEG in the result integer to indicate that an out of range conversion may have occurred.

Note 1: A possible alternative is to use the FIST64 instruction to store the converted operand to memory and access the lower 16 or 32 bits as the required integer. Even though this mechanism will not signal an attempted out of range conversion with a 16 bit or 32 bit target, it is currently in use by many compilers today.
Note 2: The values affected by this erratum are those which contain an exponent value within the affected range, AND a specific bit pattern at a specific offset within the mantissa, AND at least one non-zero bit to the right of the above bit pattern. The offset within the mantissa is a function of the floating point exponent value. The specific bit pattern is 0x8000 for FIST16 and

0x80000000 for FIST32. This means that for any given exponent within the range, one mantissa value in every 2^{16} possible mantissa values exhibits the erratum for FIST16, and one mantissa value in every 2^{32} possible mantissa values exhibits the erratum for FIST32.

Examples of affected values for FIST16, 80 bit binary notation (not an exhaustive list)
(xx means any bit pattern, yy means any non-zero bit pattern)

sign	exponent	mantissa
1	100000000010100	1xxxxx1000000000000000yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy
1	100000000010101	1xxxxxx1000000000000000yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy
1	100000000010110	1xxxxxxx1000000000000000yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy

Examples of affected values for FIST32, 80 bit binary notation (not an exhaustive list)
(xx means any bit pattern, yy means any non-zero bit pattern)

sign	exponent	mantissa
1	100000000100011	1xxxx10000000000000000000000000000yyyyyyyyyyyyyyyyyyyyyyyyy
1	100000000100100	1xxxxx10000000000000000000000000000yyyyyyyyyyyyyyyyyyyyyyyyy
1	100000000100101	1xxxxxx10000000000000000000000000000yyyyyyyyyyyyyyyyyyyyyyyyy

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

26. FLUSH# Assertion Disables L2 Machine Check Exception Reporting

PROBLEM: Upon FLUSH# assertion, the L2 Machine Check Exception generation is disabled. Once the FLUSH# pin is asserted, the processor disables the L2 MCA, by clearing the associated MCi_CTL control register to "0"'s. This operation is invisible to the software being executed.

IMPLICATION: Errors that should be reported by the L2 MCA are not reported from the time that the FLUSH# signal is asserted until the time that the MCi_CTL register is written back to all "1"s. All other errors will continue to be logged as normal.

WORKAROUND: Platform specific code (e.g., BIOS or system management software) has the potential for driving a device to assert the FLUSH# pin. If the platform specific code asserts the FLUSH# pin, this code should be enhanced to detect that MCA Exceptions are globally enabled (via register CR4.MCE). The code should then write "0"s to all of the MCi_CTL registers to clear any spurious entries and then write "1"s to all of the MCi_CTL registers in order to re-enable exception reporting. Hardware devices in systems that require L2 error reporting which could assert the FLUSH# pin should not assert FLUSH#.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

27. EFLAGS May Be Incorrect After A Multiprocessor TLB Shootdown

PROBLEM: When the Pentium II processor executes a read-modify-write arithmetic instruction with memory as the destination, it is possible for a page fault to occur during the execution of the store on the memory operand after the read operation has completed but before the write operation completes. In this case the EFLAGS value pushed onto the stack of the page fault handler may be reflective of the status of the EFLAGS register after the instruction would have completed execution rather than that before it has executed under a certain set of circumstances. This class of instruction will initially perform a load operation that has the side effect of ensuring that the final store portion of the instruction will successfully complete. The load ensures this by bringing the page table information of the page containing the data into the DTLB. This page entry could be evicted from the DTLB by speculative loads from other instructions that hit the same way of the DTLB, before the store is executed. DTLB eviction will require at least three load operations that have linear address bits 15:12 equal to each other and address bits 31:16 different from each other in close physical proximity to the arithmetic operation. If, in the very small window of time between the page eviction and the store execution, the page table entry has had its page permissions tightened (e.g., from Present to Not Present, or from Read/Write to Read Only, etc.) by the operating system in main memory by another processor (with no corresponding synchronization and subsequent TLB flush, the store will generate a DTLB miss and a call to the OS's page fault handler. The EFLAGS register may have already been updated by the arithmetic portion of the instruction before entry to the page fault handler. If under these circumstances the fault handler elects to restart the instruction the re-execution may generate an incorrect result. Instructions affected by this erratum are the memory destination forms of ADC, SBB, RCR & RCL (instructions that use a flag, carry, as input to the instruction). It should be noted that the locked version of these instructions is not impacted by this erratum.

IMPLICATION: This scenario can only occur in a multiprocessor system running under an operating system that implements a "lazy" TLB shootdown. Lazy TLB shootdown occurs when one processor makes changes to the page tables in memory, and then signals other processors to remove the page entry from their TLB without a multiprocessor synchronization being performed. To date, Intel has not observed this erratum in any laboratory testing of commercially available software applications.

In a multiprocessor system the arithmetic flags of the EFLAGS register and its memory stack image, may contain incorrect data if the read-modify-write arithmetic instruction encounters a page fault. Page Fault handler software that uses the resulting EFLAGS may see incorrect information. If the original instruction is restarted by the page fault handler, the instruction may produce incorrect results based on the prior modifications of the EFLAGS register.

WORKAROUND: Software may use the locked form of the ADC, SBB, RCR & RCL instructions to avoid this erratum. Operating systems should ensure that no processor is currently accessing a page that is scheduled to have its page permissions tightened, e.g., moved from Present to Not Present or have a page fault handler that can handle any incorrect state. Intel is working with

Multiprocessor Operating System vendors to ensure that an OS level workaround is implemented as required.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

28. Delayed Line Invalidation Issue During 2-Way MP Data Ownership Transfer

PROBLEM: In 2-way MP systems with two or more processors, each processor may attempt to modify a different portion of the same cache line, referenced as line 'A' in the discussion below. In this case (with processors noted as 'P0' and 'P1') where each processor contains a shared copy of line A in both their Level-1 and Level-2 caches, invalidation cycles must be issued by each processor prior to that processor's being able to definitively source the results of its internal write to a portion of line A to the other processors.

There exists a narrow timing window where, if for example, P0 wins the external bus invalidation race and gains ownership rights to line A due to the sequence of bus invalidation traffic, but P1 has not yet completed the pending invalidation of its own, currently valid and shared copy of line A. During that window, it is possible for a P1 internal opportunistic write to a portion of line A (while awaiting ownership rights) to occur with the original shared copy of line A still resident in P1's Level 2 cache. Such internal modification is permissible subject to delaying the broadcast of such changes until line ownership has actually been gained. However, the processor must ensure that any internal re-read by P1 of line A returns with data in the order actually written which in this case, is the data updated by the P0 write.

Specifically, the delayed line invalidation that occurs naturally due to the fact that one processor will necessarily win the invalidation race allows a narrow timing window to exist where one processor may re-read a line that it just wrote internally, but return with the data that was present from the previous shared state rather than the data updated by another processor as exhibited by the order of bus events.

IMPLICATION: Multiprocessor or threaded application synchronization that is implemented via operating system-provided synchronization constructs are not affected by this erratum. Applications which rely upon the usage of locked semaphores rather than memory ordering are also unaffected. Uniprocessor systems are not affected by this erratum. Intel has not identified to date, any commercially available application or operating system software which is affected by this erratum.

WORKAROUND: Deterministic barriers beyond which program variables will not be modified can be achieved via the usage of locked semaphore operations, and this scheme has been shown to effectively work around this erratum.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

29. Potential Early Deassertion of LOCK# during Split-Lock Cycles

PROBLEM: During a split-lock cycle there are four bus transactions; 1st ADS# (a partial read), 2nd ADS# (a partial read), 3rd ADS# (a partial write), and the 4th ADS# (a partial write). Due to this erratum, LOCK# may deassert one clock after the 4th ADS# of the split-lock cycle instead of after the RS# assertion corresponding to the 4th ADS# has been sampled. The following sequence of events are required for this erratum to occur:

1. A lock cycle occurs (split or non-split).
2. Five more bus transactions (assertion of ADS#) occur.
3. A split-lock cycle occurs and BNR# toggles after the 3rd ADS# (partial write) of the split-lock cycle. This in turn delays the assertion of the 4th ADS# of the split-lock cycle. BNR# toggling at this time could most likely happen when the bus is set for an IOQ depth of 2.

When all of these events occur, LOCK# will be deasserted in the next clock after the 4th ADS# of the split-lock cycle.

IMPLICATION: This may affect chipset logic which monitors the behavior of LOCK# deassertion.

WORKAROUND: None identified.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

30. A20M# May Be Inverted After Returning From SMM and Reset

PROBLEM: This erratum is seen when software causes the following events to occur:

1. The assertion of A20M# in real address mode.
2. After entering the 1-Mbyte address wrap-around mode caused by the assertion of A20M#, there is an assertion of SMI# intended to cause a Reset or remove power to the processor. Once in the SMM handler, software saves the SMM state save map to an area of non-volatile memory from which it can be restored at some point in the future. Then software asserts RESET# or removes power to the processor.
3. After exiting Reset or completion of power-on, software asserts SMI# again. Once in the SMM handler, it then retrieves the old SMM state save map which was saved in event 2 above and copies it into the current SMM state save map. Software then asserts A20M# and executes the RSM instruction. After exiting the SMM handler, the polarity of A20M# is inverted.

IMPLICATION: If this erratum occurs, A20M# will behave with a polarity opposite from what is expected (i.e., the 1-Mbyte address wrap-around mode is enabled when A20M# is deasserted, and does not occur when A20M# is asserted).

WORKAROUND: Software should save the A20M# signal state in non-volatile memory before an assertion of RESET# or a power down condition. After coming out of Reset or at power on, SMI# should be asserted again. During the restoration of the old SMM state save map described in event 3 above, the entire map should be restored, except for bit 5 of the byte at offset 7F18h.

This bit should retain the value assigned to it when the SMM state save map was created in event 3. The SMM handler should then restore the original value of the A20M# signal.
STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

31. Reporting of Floating-Point Exception May be Delayed

PROBLEM: The Pentium II processor normally reports a floating-point exception for an instruction when the next floating-point or MMX technology instruction is executed. The assertion of FERR# and/or the INT 16 interrupt corresponding to the exception may be delayed until the floating-point or MMX technology instruction *after* the one which is expected to trigger the exception, if the following conditions are met:

1. A floating-point instruction causes an exception.
2. Before another floating-point or MMX™ technology instruction, any one of the following occurs:
   a. A subsequent data access occurs to a page which has not been marked as accessed, or
   b. Data is referenced which crosses a page boundary, or
   c. A possible page-fault condition is detected which, when resolved, completes without faulting.
3. The instruction causing event 2 above is followed by a MOVQ or MOVD store instruction.

IMPLICATION: This erratum only affects software which operates with floating-point exceptions unmasked. Software which requires floating-point exceptions to be visible on the next floating-point or MMX technology instruction, and which uses floating-point calculations on data which is then used for MMX technology instructions, may see a delay in the reporting of a floating-point instruction exception in some cases. Note that mixing floating-point and MMX technology instructions in this way is not recommended.
WORKAROUND: Inserting a WAIT or FWAIT instruction (or reading the floating-point status register) between the floating-point instruction and the MOVQ or MOVD instruction will give the expected results. This is already the recommended practice for software.
STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

32. EFLAGS Discrepancy On A Page Fault After A Multiprocessor TLB Shootdown

PROBLEM: This erratum may occur when the Pentium II processor executes one of the following read-modify-write arithmetic instructions and a page fault occurs during the store of the memory operand: ADD, AND, BTC, BTR, BTS, CMPXCHG, DEC, INC, NEG, NOT, OR, ROL/ROR, SAL/SAR/SHL/SHR, SHLD, SHRD, SUB, XOR, and XADD. In this case, the

EFLAGS value pushed onto the stack of the page fault handler may reflect the status of the register after the instruction would have completed execution rather than before it. The following conditions are required for the store to generate a page fault and call the operating system page fault handler:

1. The store address entry must be evicted from the DTLB by speculative loads from other instructions that hit the same way of the DTLB before the store has completed. DTLB eviction requires at least three load operations that have linear address bits 15:12 equal to each other and address bits 31:16 different from each other in close physical proximity to the arithmetic operation.
2. The page table entry for the store address must have its permissions tightened during the very small window of time between the DTLB eviction and execution of the store. Examples of page permission tightening include from Present to Not Present or from Read/Write to Read Only, etc.
3. Another processor, without corresponding synchronization and TLB flush, must cause the permission change.

IMPLICATION: This scenario may only occur on a multiprocessor platform running an operating system that performs “lazy” TLB shootdowns. The memory image of the EFLAGS register on the page fault handler's stack prematurely contains the final arithmetic flag values although the instruction has not yet completed. Intel has not identified any operating systems that inspect the arithmetic portion of the EFLAGS register during a page fault nor observed this erratum in laboratory testing of software applications.
WORKAROUND: No workaround is needed upon normal restart of the instruction, since this erratum is transparent to the faulting code and results in correct instruction behavior. Operating systems may ensure that no processor is currently accessing a page that is scheduled to have its page permissions tightened or have a page fault handler that ignores any incorrect state.
STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

1AP. APIC Access to Cacheable Memory Causes SHUTDOWN

PROBLEM: APIC operations which access memory with any type other than uncacheable (UC) are illegal. If an APIC operation to a memory type other than UC occurs and Machine Check Exceptions (MCEs) are disabled, the processor will enter SHUTDOWN after such an access. If MCEs are enabled, an MCE will occur. However, in this circumstance, a second MCE will be signaled. The second MCE signal will cause the Pentium II processor to enter SHUTDOWN.
IMPLICATION: Recovery from a PIC access to cacheable memory will not be successful. Software that accesses only UC type memory during APIC operations will not encounter this erratum.
WORKAROUND: Ensure that the memory space to which PIC accesses can be made is marked as type UC (uncacheable) in the memory type range registers (MTRRs) to avoid this erratum.

STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

2AP. 2-Way MP Systems May Hang Due to Catastrophic Errors During BSP Determination

PROBLEM: In 2-way MP systems, a catastrophic error during the bootstrap processor (BSP) determination process should cause the assertion of IERR#. If the catastrophic error is due to the APIC data bus being stuck at electrical zero, then the system hangs without asserting IERR#.
IMPLICATION: 2-way MP systems may hang during boot due to a catastrophic error. This erratum has not been observed to date in a typical commercial system, but was found during focused system testing using a grounded APIC data bus.
WORKAROUND: None identified at this time.
STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

3AP. Write to Mask LVT (Programmed as EXTINT) Will Not Deassert Outstanding Interrupt

PROBLEM: If the APIC subsystem is configured in Virtual Wire Mode implemented through the local APIC, (i.e., the 8259 INTR signal is connected to LINT0 and LVT1's interrupt delivery mode field is programmed as EXTINT), a write to LVT1 intended to mask interrupts will not deassert the internal interrupt source if the external LINT0 signal is already asserted. The interrupt will be erroneously posted to the Pentium II processor despite the attempt to mask it via the LVT.
IMPLICATION: Because of the masking attempt, interrupts may be generated when the system software expects no interrupts to be posted.
WORKAROUND: Software can issue a write to the 8259A interrupt mask register to deassert the LINT0 interrupt level, followed by a read to the controller to ensure that the LINT0 signal has been deasserted. Once this is ensured, software may then issue the write to mask LVT entry 1.
STATUS: For the steppings affected see the Summary Table of Changes at the beginning of this section.

DOCUMENTATION CHANGES

The Documentation Changes listed in this section apply to the *Pentium® II Processor* datasheet. All Documentation Changes will be incorporated into a future version of the appropriate Pentium II processor documentation.

1. Note 4 of Table 13 and Note 6 of Table 16, Output Valid Delay Specified to 2.5V +5%

The following change is to Section 2.13, Table 13, *System Bus AC Specifications (CMOS Signal Group)* and Section 2.13, Table 16, *System Bus AC Specifications (TAP Connection).*
Note 4 of Table 13 and Note 6 of Table 16 in the *Pentium® II Processor* datasheet state that the 2.5V Output Valid Delay is specified to V_{CCCORE}. This is incorrect. Each note should read "Valid delay timings for these signals are specified to 2.5V +5%. See Table 3 for pull-up resistor values."

2. Invalid Arithmetic Operations and Masked Responses to Them Relative to FIST/FISTP Instruction

The *Pentium® Pro Family Developer's Manual, Volume 2: Programmer's Reference Manual,* Table 7-20 and the *Intel Architecture Software Developer's Manual, Volume 1,* Table 7-20 show "Invalid Arithmetic Operations and the Masked Responses to Them." The table entry corresponding to the FIST/FISTP condition is missing, and is shown below:

Condition	Masked Response
FIST/FISTP instruction when input operand <> MAXINT for destination operand size.	Return MAXNEG to destination operand.

When FIST/FISTP instruction is executed with input operand <> and the destination operand size is MAXINT, the floating-point zero-divide exception will return MAXNEG to the destination operand as its masked response.

3. FIDIV/FIDIVR m16int Description

The *Pentium® Pro Family Developer's Manual, Volume 2: Programmer's Reference Manual,* pages 11-118 and 11-121 and the *Intel Architecture Software Developer's Manual, Volume 1,* pages 3-118 and 3-122 show in the Description column for the FIDIV *m16int* instruction as "Divide ST(0) by *m64int* by ST(0) and store the result in ST(0)" and FIDIVR *m16int* instruction as "Divide *m64int* by ST(0) and store the result in ST(0)" In both of these cases, *m64int* should be replaced with *m16int*.

SPECIFICATION CLARIFICATIONS

The Specification Clarifications listed in this section apply to the *Pentium® II Processor at 233 MHz, 266 MHz and 300 MHz* datasheet. All Specification Clarifications will be incorporated into a future version of the appropriate Pentium II processor documentation.

1. Writes to WC Memory

Section 11-3 of the *Pentium® Pro Family Developer's Manual, Volume 3: Operating System Writer's Guide*, identifies that "Writes" to a region of WC memory "may be delayed and combined in the write buffer to reduce memory accesses." This sentence should state that "Writes" to a region of WC memory "may be delayed and combined in the write buffer to reduce memory accesses. The writes may be delayed until the next occurrence of a buffer or processor serialization event, e.g., CPUID execution, a read or write to uncached memory, interrupt occurrence, LOCKed instruction execution etc. if the WC buffer is partially filled."

2. Multiple Processors Protocol and Restrictions

Section 7.5.2., *Protocol Requirements and Restrictions*, in the *Pentium® Pro Family Developer's Manual, Volume 3: Operating System Writer's Guide*, contain inconsistencies which will be clarified as follows:

7.5.2. Protocol Requirements and Restrictions

The MP protocol imposes the following requirements and restrictions on the system:

- An APIC clock (APICLK) must be provided on all systems based on the Pentium® Pro processor.
- All interrupt mechanisms must be disabled for the duration of the MP protocol algorithm including the window of time between the assertion of INIT# or receipt of an INIT IPI by the application processors and the receipt of a STARTUP IPI by the application processors. That is, requests generated by interrupting devices must not be seen by the local APIC unit (on board the processor) until the completion of the algorithm. Failure to disable the interrupt mechanisms may result in processor shutdown.
- The MP protocol should be initiated only after a hardware reset. After completion of the protocol algorithm, a flag is set in the APIC base MSR of the BSP (APIC_BASE.BSP) to indicate that it is the BSP. This flag is cleared for all other processors. If a processor or the system is subject to an INIT sequence (either through the INIT# pin or an INIT IPI), then the MP protocol is not re-executed. Instead, each processor examines its BSP flag to determine whether the processor should boot or wait for a STARTUP IPI.

3. NMI Handling While in SMM

Section 9.7., *NMI Handling While in SMM*, in the *Pentium® Pro Family Developer's Manual, Volume 3: Operating System Writer's Guide*, will be clarified as follows:

9.7 NMI Handling while in SMM

NMI interrupts are blocked upon entry to the SMI handler. If an NMI request occurs during the SMI handler, it is latched and serviced after the processor exits SMM. Only one NMI request will be latched during the SMI handler. If an NMI request is pending when the processor executes the RSM instruction, the NMI is serviced before the next instruction of the interrupted code sequence.

Although the NMI requests are blocked when the CPU enters SMM, they may be enabled through software by executing an IRET/IRETD instruction. If the SMM handler requires the use of NMI interrupts, it should invoke a dummy interrupt service routine for the purpose of executing an IRET/IRETD instruction. Once an IRET/IRETD instruction is executed, NMI interrupt requests are serviced in the same "real mode" manner in which they are handled outside of SMM.

4. Critical Sequence of Events During a Page Fault Exception

Section 3.6.4., *Page-Directory and Page-Table Entries*, in the *Pentium® Pro Family Developer's Manual, Volume 3: Operating System Writer's Guide*, will be clarified as follows:

If the processor generates a page-fault exception, the operating system must carry out the following operations in this order:

1. Copy the page from disk storage into physical memory if needed.
2. Load the page address into the page-table or page-directory entry and set its *present* flag. Other bits, such as the dirty and accessed bits, may also be set at this time.
3. Invalidate the current page table entry in the TLB (see Section 3.7., *Translation Lookaside Buffers (TLBs)*, for a discussion of TLBs and how to invalidate them).
4. Return from the page fault handler to restart the interrupted program or task.

SPECIFICATION CHANGES

The Specification Changes listed in this section apply to the *Pentium® II Processor* datasheet. All Specification Changes will be incorporated into a future version of the appropriate Pentium II processor documentation.

1. BCLK Period Stability Change from ±250ps to ±300ps

The following change is to Section 2.13, Table 10, *System Bus AC Specifications (Clock).* The BCLK Period Stability for the Pentium II processor is changed from ±250 picoseconds (ps) to ±300ps. This will be incorporated into the next revision of the *Pentium® II Processor* datasheet, Table 10, Section 2.13. The following reflects the exact changes to the table:

Table 10. System Bus AC Specifications (Clock)

T# Parameter	Min	Nom	Max	Unit	Figure	Notes
T2: BCLK Period Stability			**±300**	ps		7, 8

NOTES:

7. Due to the difficulty of accurately measuring processor clock jitter in a system, it is recommended that a clock driver be used that is designed to meet the period stability specification into a test load of 10 to 20 pF. This should be measured on the rising edges of adjacent BCLKs crossing 1.25V. The jitter present must be accounted for as a component of BCLK timing skew between devices.
8. The clock driver's closed loop jitter bandwidth must be set low to allow any PLL-based device to track the jitter created by the clock driver. The –20 dB attenuation point of the clock driver, as measured into a 10 to 20 pF load, should be less than 500 kHz. This specification may be ensured by design characterization and/or measured with a spectrum analyzer.

2. 300MHz Pentium® II Processor Vcc_{CORE} Transient Tolerance Change by 5mV and 10mV

The following changes are to Section 2.13, Table 6, *Pentium® II Processor Voltage and Current Specifications*.
The Baseboard Transient Tolerance in Table 6 is changed from minimum of -0.150V and maximum of 0.150V to a minimum of -0.145V and a maximum of 0.145V.
The Vcc_{CORE} Transient Tolerance is also changed from a minimum of -0.195V and a maximum of 0.195V to a minimum of -0.185V and a maximum of 0.185V. The following reflects the exact changes to the table:

This is a 300 MHz Pentium II processor only change.

Table 6. Pentium® II Processor Voltage and Current Specifications

Symbol	Parameter	Core Freq	Min	Typ	Max	Unit	Notes
Baseboard Tolerance, Transient	Baseboard voltage, transient tolerance level	**300 MHz**	**-0.145**		**0.145**	V	5
V_{ccCORE} Tolerance, Transient	V_{ccCORE} voltage, transient tolerance level	**300 MHz**	**-0.185**		**0.185**	V	6

NOTES:

5. These are the tolerance requirements, across a 20 MHz bandwidth, **at the Slot 1 connector pins on the bottom side of the baseboard**. The requirements at the Slot 1 connector pins account for voltage drops (and impedance discontinuities) across the connector, substrate edge fingers and to the processor core. The Slot 1 connector has the following requirements: Pin Self Inductance: 10.5 nH(max); Pin to Pin Capacitance: 2 pF(max, at 1 MHz); Contact Resistance: 12 mΩ (max averaged over power/ground contacts). Contact Intel for testing conditions of these requirements.
6. These are the tolerance requirements, across a 20 MHz bandwidth, **at the processor substrate edge fingers**. The requirements at the processor substrate edge fingers account for voltage drops (and impedance discontinuities) at the substrate edge fingers and to the processor core.

3. Deep Sleep ICC CORE Specification

In the *Pentium® II Processor at 233 MHz, 266 MHz and 300 MHz* datasheet, the ICC_{DSLP} CORE MAX parameter, Table 6 on page 19 has been changed. Specifically, ICC_{DSLP} CORE MAX has been changed from 0.2 A to 0.35 A.

4. TRST# Pull-down Resistor

The Pentium II processor has a circuit to automatically reset the TAP port during a power on cycle. This circuit may not properly function during certain Vcc_{CORE} supply ramp up and ramp down conditions if a pull-up resistor is used on TRST#. If the TAP port is not properly reset the processor may enter a test mode and fail to issue an instruction fetch. In the *Pentium® II Processor at 233 MHz, 266 MHz and 300 MHz* datasheet, Table 3, page 15, Note 3, a 470 ohm pull-down resistor is recommended on TRST#. This has been changed to read "The TRST# signal must be driven low at power on reset. This can be accomplished with a 680 ohm pull-down resistor". This value has been changed to reflect the reduced current drive capability of the ITP565. Also, Appendix Section A.1.52 currently reads "Pentium II processors self-reset during power on; therefore, it is not necessary to drive this signal during power on reset". This has been changed to read "TRST# must be driven low during power on reset. This can be accomplished with a 680 ohm pull-down resistor".

5. 300 MHz Pentium® II Processor Maximum Temperature Change

The following change is to Section 4.1, Table 20, page 38 of the *Pentium® II Processor at 233 MHz, 266 MHz, and 300 MHz* datasheet. The Max Tplate and Max Tcover specifications for the 300 MHz Pentium II processor have been increased from 70°C to 72°C.

Table 20. Pentium® II Processor Thermal Design Specification[1]

Processor Core Frequency (MHz)	L2 Cache Size (kB)	Max Processor Power[2](W)	Max Thermal Plate Power[3] (W)	Min T_{PLATE} (°C)	Max T_{PLATE} (°C)	Min T_{COVER} (°C)	Max T_{COVER} (°C)
300	512	43.0	41.4	5	**72**	5	**72**

6. S.E.C. Cartridge Mechanical Specifications

The S.E.C. cartridge mechanical specifications provided in the *Pentium® II Processor at 233 MHz, 266 MHz and 300 MHz* datasheet contained incorrect tolerance information for some dimensions. Also, some of the information provided in the figures was duplicitous and not required to accurately use and design for the S.E.C. cartridge package. The revised figures provided in this section are meant to replace Figures 29 through 42, Table 23 and the accompanying notes found on page 44.

NOTES FOR FIGURE 1 THROUGH FIGURE 11

Unless otherwise specified, the following drawings are dimensioned in inches. All dimensions provided with tolerances are guaranteed to be met for all normal production product.

Figures and drawings labeled as “Reference Dimensions” are provided for informational purposes only. Reference Dimensions are extracted from the mechanical design database and are nominal dimensions with no tolerance information applied. Reference Dimensions are NOT checked as part of the processor manufacturing.

Drawings are not to scale.

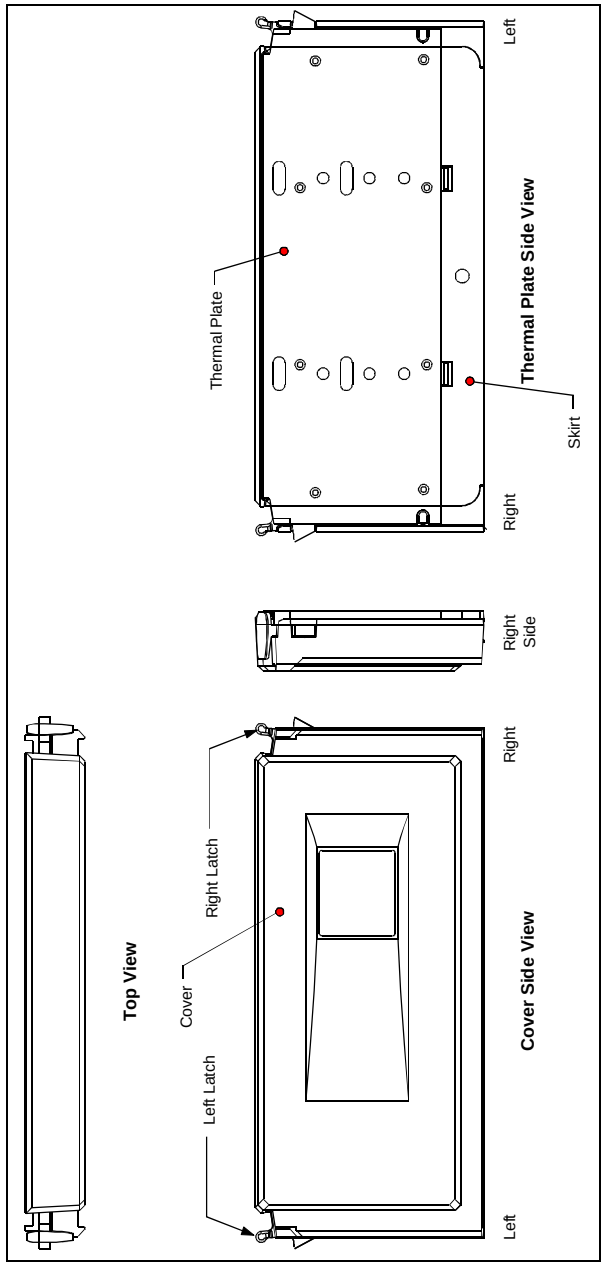


Figure 1. S.E.C. Cartridge Top and Side Views

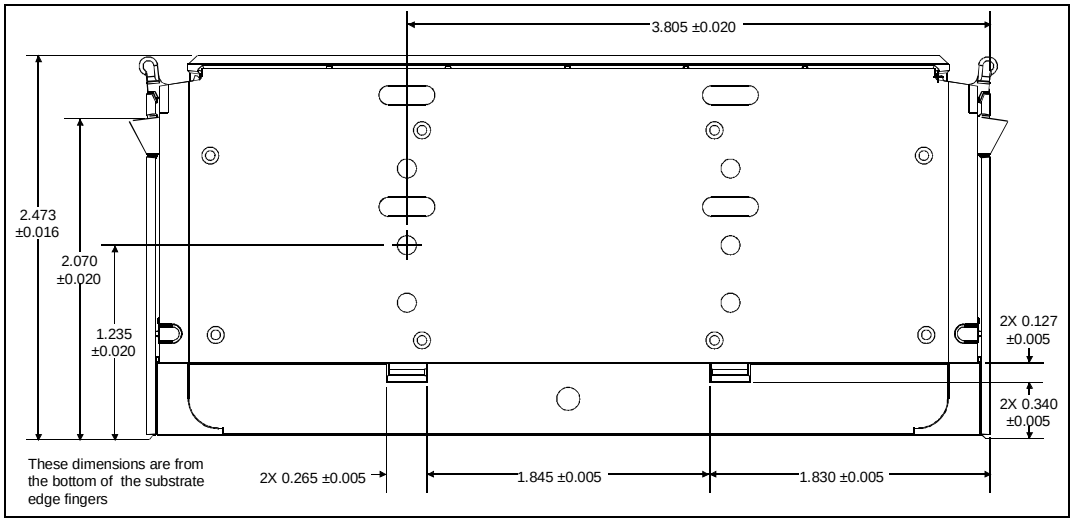


Figure 2. S.E.C. Cartridge Thermal Plate Side Dimensions

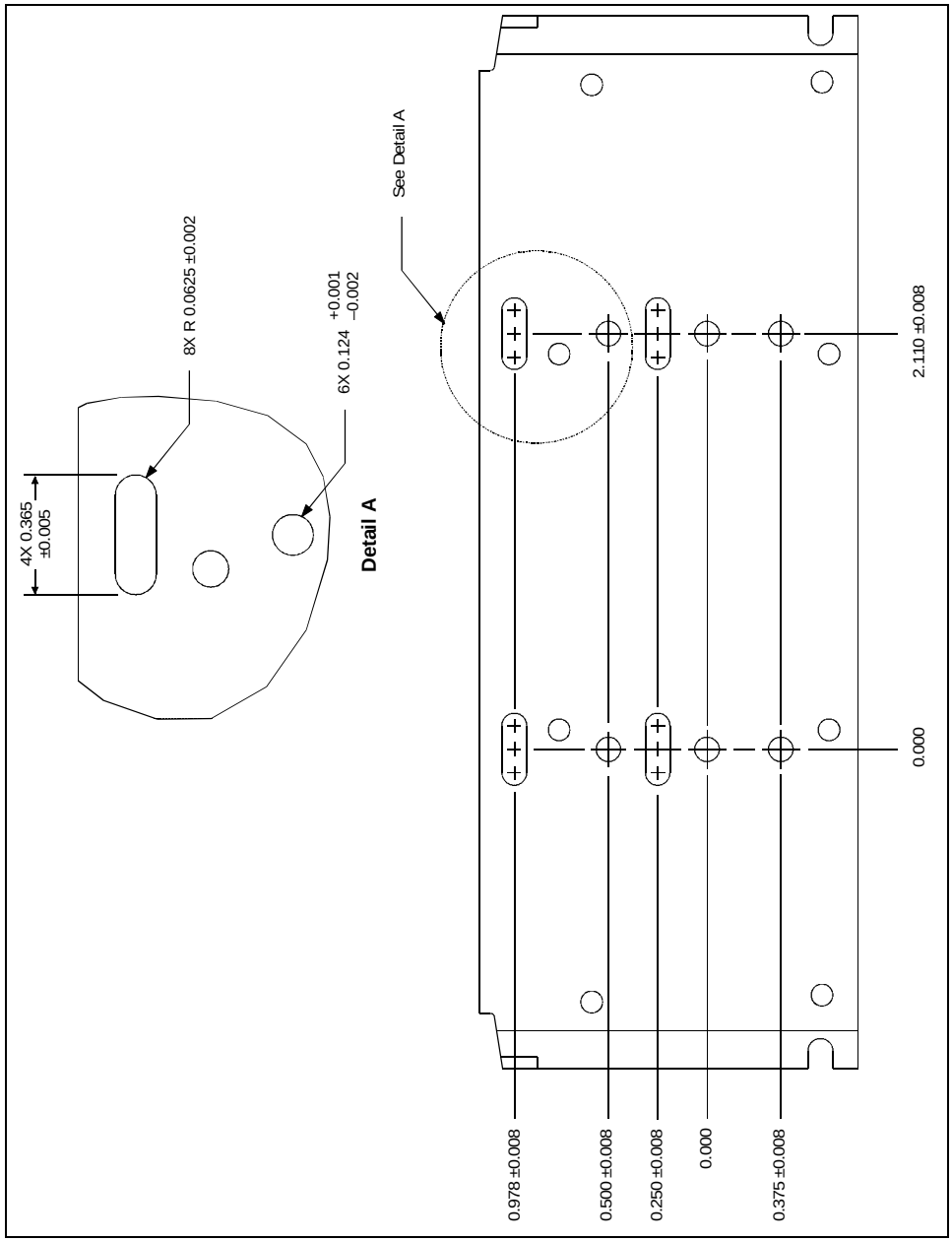


Figure 3. S.E.C. Cartridge Thermal Plate Side Dimensions

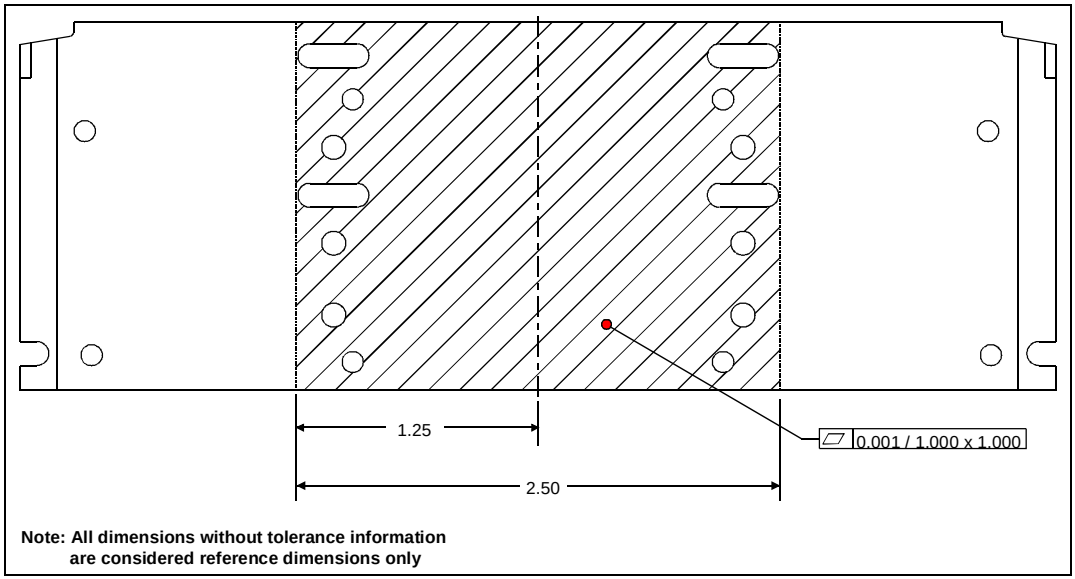


Figure 4. S.E.C. Cartridge Thermal Plate Flatness Dimensions

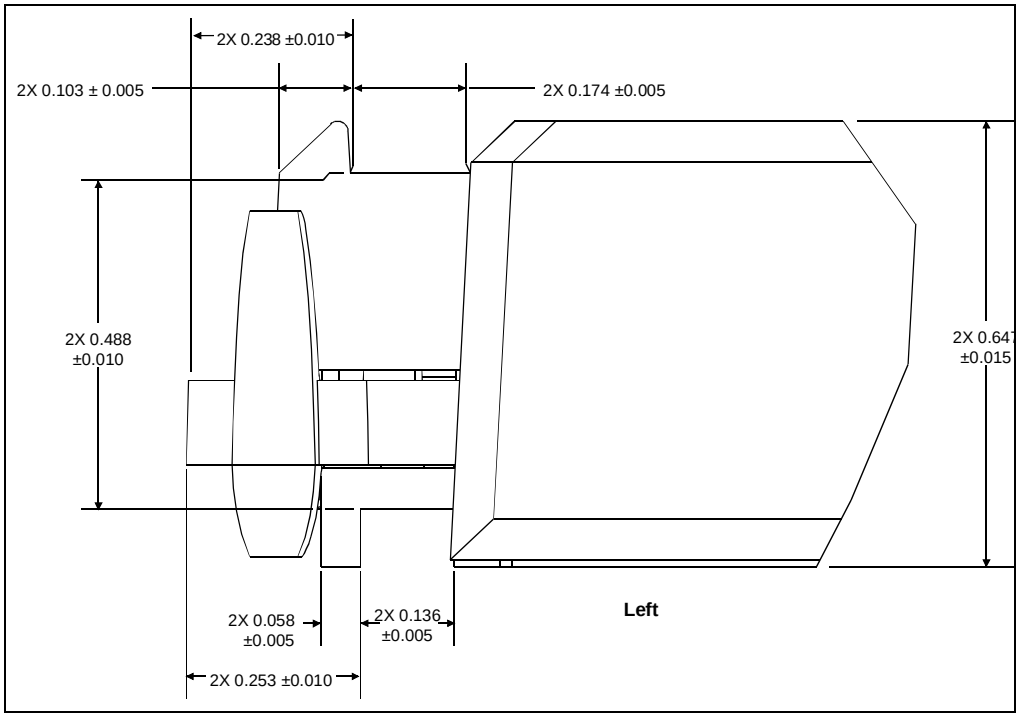


Figure 5. S.E.C. Cartridge Latch Arm, Thermal Plate Lug and Cover Lug Dimensions

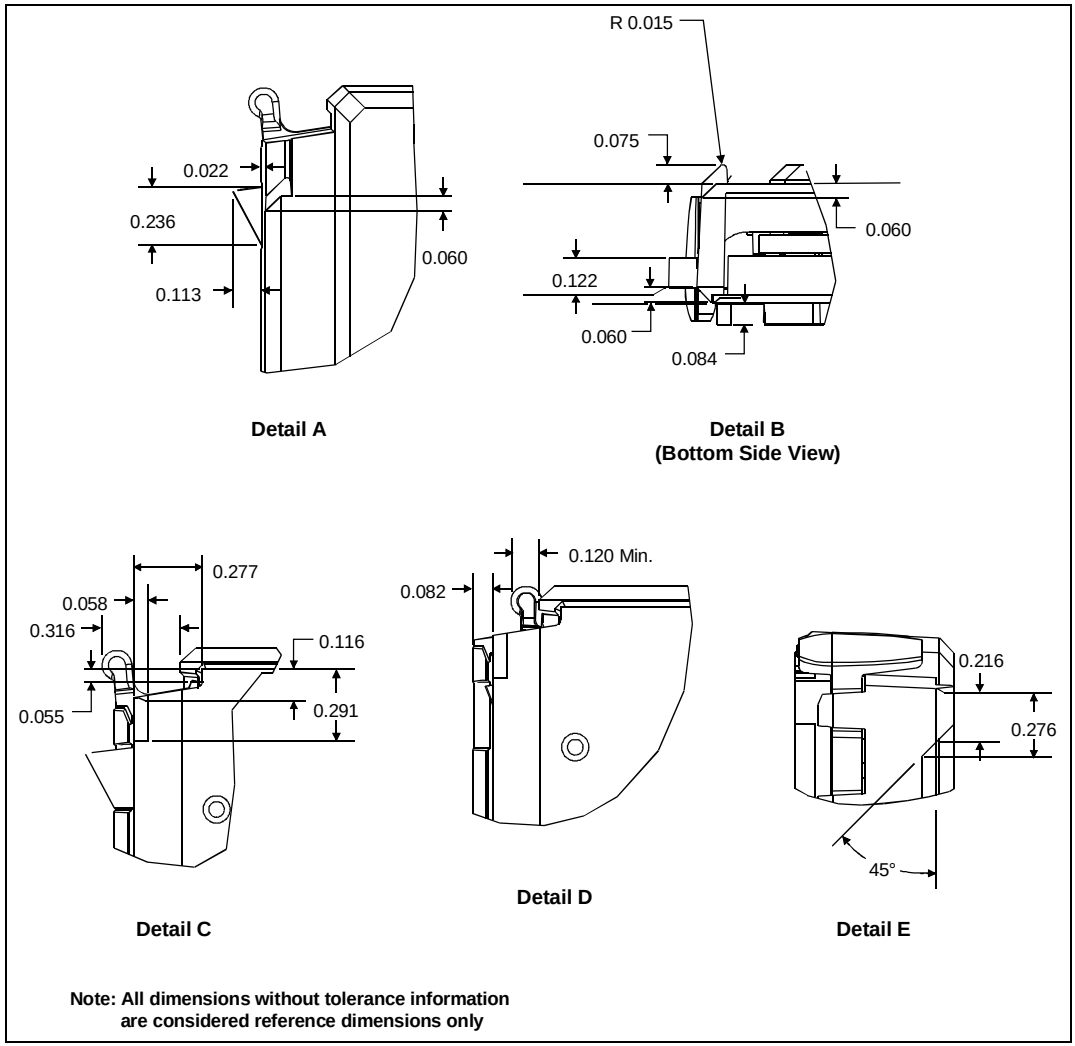


Figure 6. S.E.C. Cartridge Latch Arm, Cover and Thermal Plate Detail Dimensions

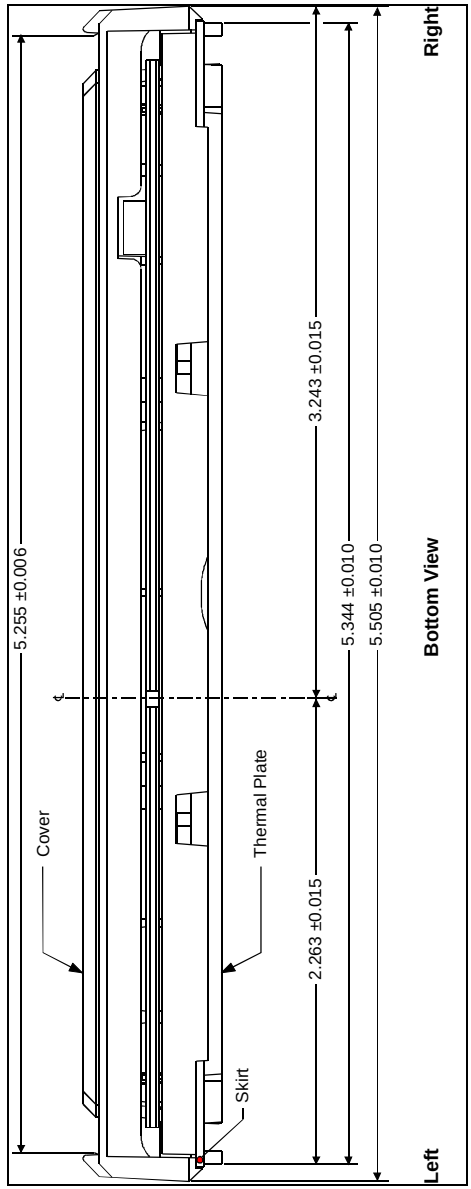


Figure 7. S.E.C. Cartridge Bottom Side View

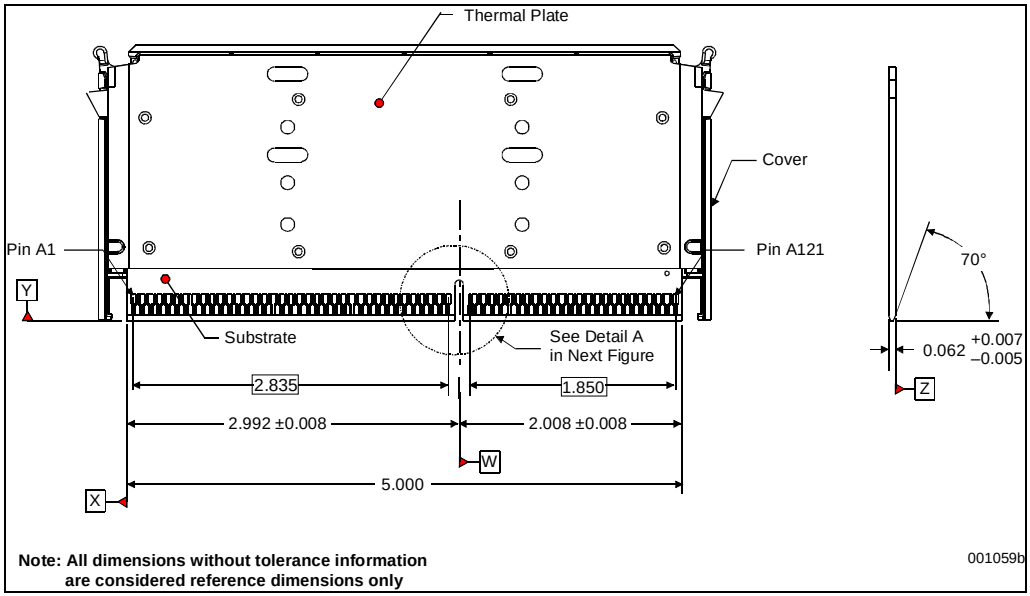


Figure 8. S.E.C. Cartridge Substrate Dimensions (Skirt not shown for clarity)

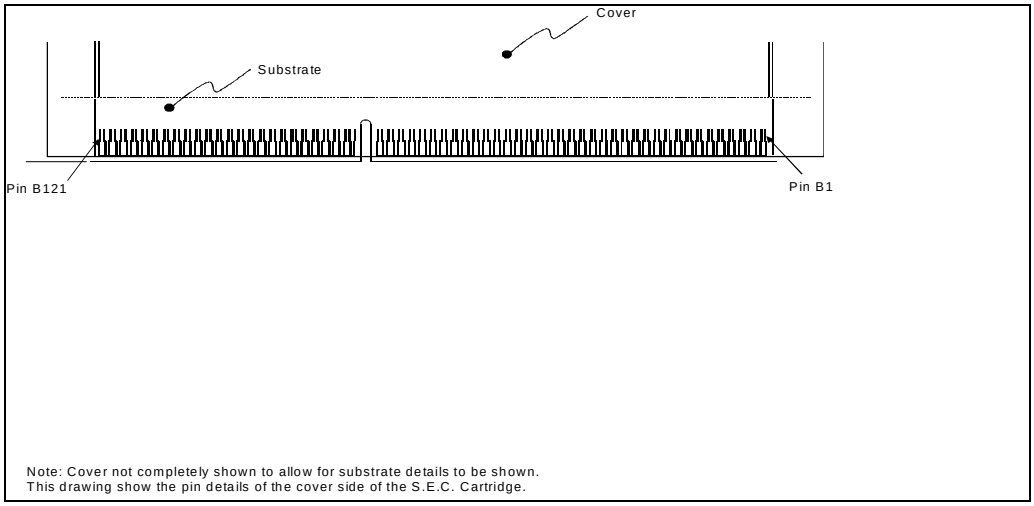


Figure 9. S.E.C. Cartridge Substrate Dimensions, Cover Side View

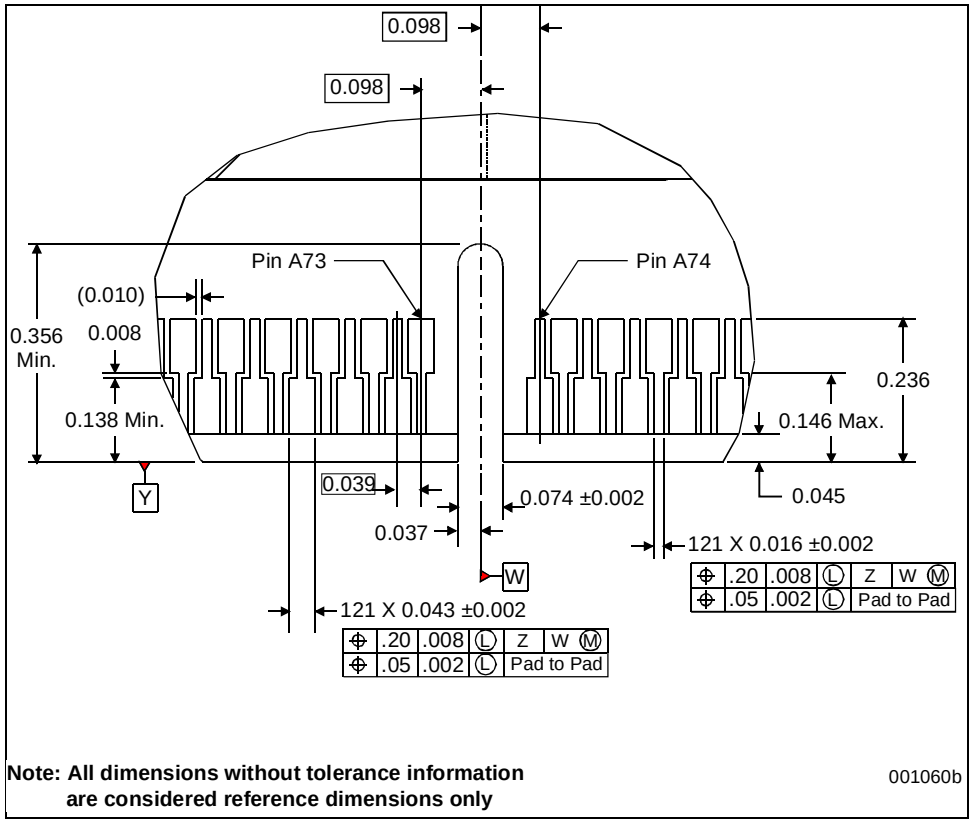


Figure 10. S.E.C. Cartridge Substrate Detail A

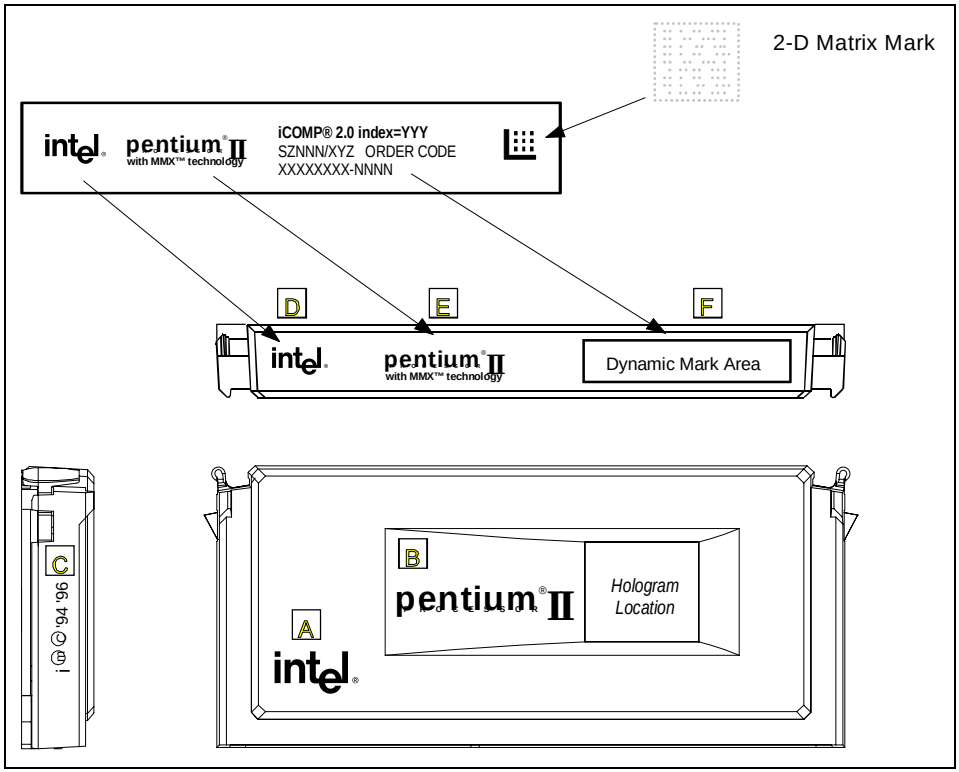


Figure 11. S.E.C. Cartridge Mark Locations
(See Table 1 For Processor Markings)

Table 1. Description Table for Processor Markings

Code Letter	Description
A	Logo
B	Product Name
C	Trademark
D	Logo
E	Product Name
F	Dynamic Mark Area – with 2-D matrix

7. *S.E.C. Cartridge Heatsink Attach Clearance Specification Change*

A change to the attach mechanism specification is provided below. The original specification for clearance of heatsink attach solutions is provided in Section 4.3 (and Figure 23 and Figure 24) in the *Pentium® II Processor at 233 MHz, 266 MHz and 300 MHz* datasheet. Figure 11 shows the revised maximum specifications for heatsink clip and Rivscrew* attach mechanism designs. All heatsink clip and Rivscrew attachment solutions must remain within the respective maximum "permissable" zone as detailed in Figure 11.

The thermal plate shown in Figure 11 is used to portray the heatsink attach mechanism design space for the future Slot 1 processors. However, the maximum "permissable" design space is also valid for both the current Pentium II and future Slot 1 processors.

Please review the proposed specifications to guarantee compatibility with existing Pentium II and future Slot 1 processor heatsink attachment solutions. Note that the required heatsink attachment solution space should not exceed the maximum proposed design space while installed or during the installation and removal procedure.

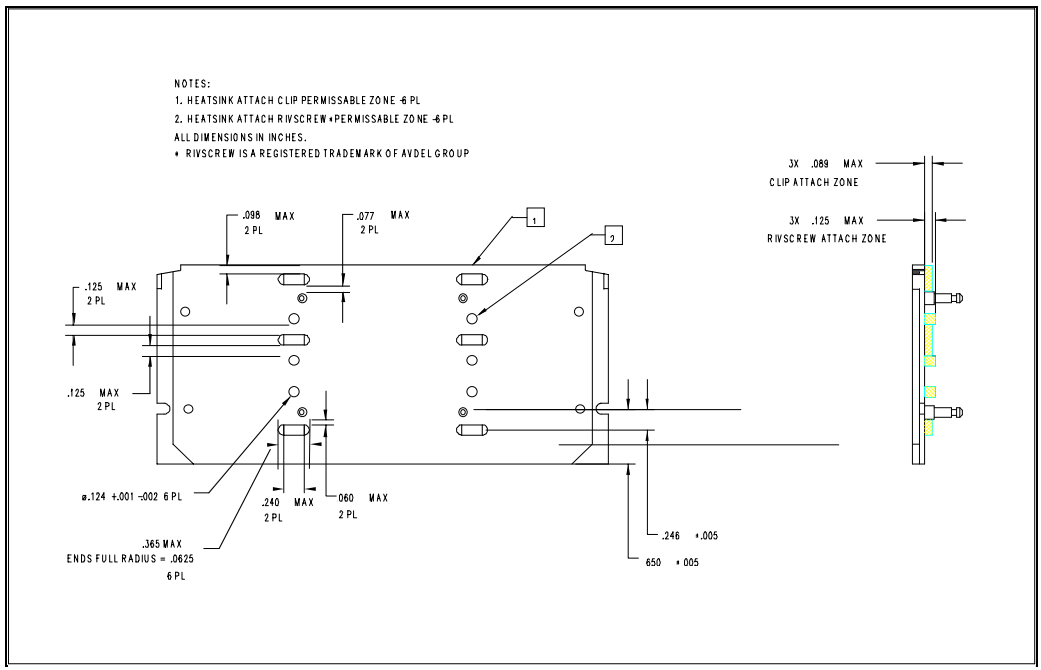


Figure 12. Heatsink Attachment Mechanism Design Space